CSCI651 Project #1

The goal of this project is to familiarize with the socket programming API and design a protocol with client, server, and proxy server applications. The project is expected to be completed on an individual basis with your own proprietary protocol; there is no requirement to interoperate with another classmate's application.

Application Requirements:

The application shall provide the user the ability to query a "clock" server and provide the user the current configured "time" expressed in either Unix epoch time or Coordinated Universal Time (UTC) formats. A multi-threaded proxy server will sit between clients and the origin time server to handle simultaneous requests from multiple clients. A client with the appropriate credentials shall be able to modify the "clock" on the server. The "time" on the server is static and only changes if a client modifies the value.

The user shall receive a report via standard output of the delay to receive an answer. The report shall provide a breakdown of the delay for each hop through a proxy server (if any) and original server in the network. No specific report/output format is specified.

System Requirements:

Application will be written using Java, C, or C++ programming language. The application should run on Ubuntu 14.04.3 LTS. The JRE installed on the test system is Java 8.

Submission Requirements:

Submissions without all items listed below will receive a failing grade.

1. Documentation on the application protocol messages, formats, error codes, etc. including example session flows for success and failure cases. The documentation should resemble an RFC in format/detail of the protocol.
2. Application source code. Source code should be well organized and clearly documented using an acceptable standard (such as Doxygen comments).
3. Pre-compiled binaries compatible with system requirements.
4. Submit a single archive (zip/tar/tar.gz/rar) containing the following directory structure:

/mhvcs	- top level directory (no files)
/mhvcs/src	- *.java/*.c/*.cpp files for your program
/mhvcs/doc	- readme/documentation files
/mhvcs/bld	- *.class/*.o/*.exe (binaries/executable per language)

Substitute MY RIT username (mhvcs) with YOUR RIT username.

Command Line Interface Requirements:

Rather than distinct binaries, a single binary will present the user with the modes of operation and associated options. Ignoring any language specific context, such as `java`, the application shall be invoked from the command line as follows:

```
tsapp -{c,s,p} [options] [server address] port [2nd port]
```

Command line options: ({} are required, [] are optional based on use case/application). Individual `options` may appear in any order.

`–c`: run as client

`–s`: run as server

`–p`: run as proxy

`–u`: use UDP. *client & proxy server applications*

`–t`: use TCP. *client & proxy server applications*

`–z`: use UTC time. *client applications.*

`–T <time>`: set server time. *client & server applications.*

`--user <name>`: credentials to use. *client & server applications*

`--pass <password>`: credentials to use. *client & server applications.*

`-n <#>`: number of consecutive times to query the server. *client applications.*

--proxy-udp <#>: server UDP port to use. *proxy application*

--proxy-tcp <#>: server TCP port to use. *proxy application*

`server address`: address of server to connect to. *client & proxy server applications*.

`port`: primary connection port. *server & proxy applications use this as UDP receiving port. client uses this as destination port for both UDP/TCP based on* –u*.*

2nd `port`: alt. connection port. *server & proxy use this as TCP listening port.*

Grading/Evaluation:

Projects will be evaluated in accordance with all requirements sections. Projects will be executed using a script which will capture all output of the application; the output of the application as captured will be used to determine the implementation functions as required. This output will be provided as a record and points will be earned as follows:

60 pts: Use cases (four) provided ahead of time

20 pts: Protocol Documentation

10 pts: Use cases not provided ahead of time

10 pts: Report delivered to user with delay/hop breakdown

Projects not adhering to Submission Requirements *will be assessed a 50% penalty of any earned points.*

Hints:

Consider first the application protocol: What messages does the client need to send to the server? Server to client? What information does each message need to contain and how will it be represented?

Use an iterative approach to implementation: First implement a basic client-server model using UDP, then add TCP, add RTT calculation, etc. Finally, implement the proxy server.

Review course material in the text book and study well known command line applications such as `ping`, `tracert`, and `iperf`.

Do not wait until the final week to complete this assignment.